- Vad är Javascript
  - Javascript är ett front-end språk
  - Avsluta kod med semikolon
  - Kommentarer
  - Stor Bokstav är inte samma som liten bokstav
  - Inspektera en hemsida
  - Vad gör man när man fastnar
  - Frivillig extra läsning
- Variablar
  - Vad är en variabel
  - Skapa en variabel
  - Variabel namn
  - Använda en variabel
  - Ändra en variabel
  - Variabel typer
  - Frivillig extra läsning
- Funktioner
  - Vad är en funktion?
  - Skapa en funktion
  - Använda en funktion
  - Parameter
  - Javascript standard funktioner
  - Frivillig extra läsning
- Integer(nummer)
  - Aritmetik
  - Frivillig extra läsning
- String(text)
  - Vad är en string
  - Hur många tecken är min string
  - Sätta ihop flera strings
  - Javascripts string funktioner
  - Frivillig extra läsning
- If-Else
  - Vad är if-else
  - Jämföra värde
  - Else if
  - Logical Operators
  - Frivillig extra läsning

- Arrays
  - Vad är en array
  - Använda arrays
  - Array funktioner
  - Array length
  - Frivillig extra läsning
- Loops
  - Vad är en loop
  - for loop
  - while loop
  - Frivillig extra läsning
- Javascript och HTML
  - HTML
  - Börja använda javascript på riktigt
  - Hitta html element
  - Ändra html element
    - Ändra Stil
  - Frivillig extra läsning
- Events
  - Vad är Events
  - Kod Exempel
  - Frivillig extra läsning

Vad är Javascript

Javascript är ett programmeringsspråk som används av (nästan) alla hemsidor i världen.
Javascript kommer med i webbläsare så man behöver inte installera det.
Javascript används tillsammans med HTML och CSS för att skapa hemsidor. Man kan säga att HTML lägger strukturen på hemsidan, CSS gör sidan fin och Javascript gör att man kan interagera med hemsidan.
Alla programmeringsspråk ser ganska lika ut men har vissa skillnader, om man kan ett språk så har man en bra start för att lära sig andra. Några saker som alla språk använder sig av är till exempel variabler, loopar, villkor, händelse, operator, funktioner och mer. Alla dessa förklarar jag i detta dokument.

Javascript är ett front-end språk

Man kan dela upp programmeringsspråk i front-end eller back-end.

Backend språk används av en server som ofta är kopplat till en databas. Den hämtar data från databasen och skickar den till klienten(den som använder hemsidan eller programmet). Om du är på en sida så har du inte tillgång till backend-koden, den finns på en server av den som äger hemsidan.

Frontend språk är inte kopplade till databasen, den får data från backend och visar det till användaren. Ett frontend språk ligger inte på servern, den ligger på användarens dator. När du använder en hemsida så är det din dator som har fått en javascript fil och läser den, du har full tillgång till javascript koden.

Det är faktiskt möjligt att använda javascript som backend språk med något som heter node.js, men det är överkurs.

Avsluta kod med semikolon

I Javascript så avslutar man en bit kod med ett semikolon(;). Detta berättar för datorn att denna delen av koden är klar och nästa del kommer börja. Det är väldigt vanligt att när man har problem i javascript så har man glömt ett semikolon så se upp efter det om koden inte funkar.

Kommentarer

I din Javascript kod kan du lämna kommentarer. En kommentar kommer inte att köra någon kod utan är bara för den som läser din kod och vill förstå vad som händer. Det finns två sätt att skriva kommentarer antigen med två snedstreck
// Detta är en kommentar
Eller omringa med ett snedstreck och en stjärna
/*
Detta är en kommentar
Detta är fortfarande en kommentar
***/**

Stor Bokstav är inte samma som liten bokstav

Javascript är ett språk som är “case sensitive", det betyder att ett namn som har stora bokstäver kommer vara annorlunda från samma namn med små bokstäver. Därför är det viktigt att hålla koll på små och stora bokstäver i din kod.
Till exempel: **Hejsan** är inte samma som **hejsan**

Inspektera en hemsida

Något du kanske känner till är att på alla sidor kan du högerklicka och sedan trycka på “inspect”, detta öppnar ett nytt fönster på höger sida. Här kan du se mycket information om en sida och det är uppdelat i olika tabbar(elements, console, sources, osv).

Elements är där du kan se html koden för den sidan vilket ofta kan vara användbart. Du kan hitta id och klasser för olika html element som du sedan kan använda javascript för att ändra.
Console är där du kan hitta felmeddelande om det finns några eller om det är fel på din javascript kod. Det är också möjligt att skriva ut information till denna konsol vilket är ofta användbart. Koden för det ser ut så här: Console.log(“Detta skrivs ut i konsolen”);
Sources är där du kan se all front-end kod. Detta inkluderar HTML, CSS, Javascript, bilder, mm. Det kan vara bra om du vill lära dig från någon annans Javascript kod.

Vad gör man när man fastnar

Alla programmerare har tider då någonting går fel och man håller på att bli galen. Det finns flera sätt att försöka fixa detta. Här är några länkar som kan hjälpa i sådana situationer:

- www.google.com - Det är väldigt vanligt att använda google för att söka på problemet
- https://stackoverflow.com/ - När man söker på google är det stor chans att man hamnar på denna hemsida, det är en sida för programmerare att ställa frågor och få svar
- https://chatgpt.com/ - På stående har AI blivit ett stort ämne, en av sakerna AI är bäst på är programmering. Skriv en fråga precis som du hade frågat en människa och var inte rädd att ställa uppföljningsfrågor
- https://www.w3schools.com/js/ - Denna sida har mycket bra info om javascript och på vänster sida kan du välja den exakta delen av javascript du har problem med för att lära dig mer.

Om ingenting annat fungerar så rekommenderar jag att ta en paus och göra någonting annat ett tag så din hjärna kan jobba på det i bakgrunden, när du kommer tillbaka kanske du vet exakt hur du ska göra.

Frivillig extra läsning

Här kan du gå djupare in i detta kapitlet om du vill:

https://en.wikipedia.org/wiki/JavaScript

https://developer.mozilla.org/en-US/docs/Web/JavaScript

https://www.w3schools.com/js/

Variablar

Vad är en variabel

En variabel är en bit data(text, nummer, osv) som man har gett ett namn för att kunna använda senare. En variabel består av 3 delar, **typ**, **namn** och **data**.
I andra programmeringsspråk är det vanligt att man behöver säga vilken typ av data det är, i javascript behöver man inte det för javascript vet vilken typ det är automatiskt. Man använder istället ordet **let**, ibland kan man använda **var** eller **const** men det är överkurs för nu.

Skapa en variabel

Här är ett exempel på att skapa en javascript variabel:
let myNumber = 15;
I detta exempel är namnet “myNumber” och datan är 15. Typen är nummer vilket brukar kallas **int**, men vi behöver inte säga typen i javascript, vi använder **let** istället.

Variabel namn

Det är bra att veta vad man kan ge en variabel för namn. Namnet kan innehålla bokstäver, siffror, understreck, och dollartecken, men kan inte börja på en siffra. Det finns också vissa namn som javascript redan använder så du kan inte använda dem, ett exempel på detta är **let**

Exempel på korrekta variabel namn:

- mittnummer
- myVariable123
- thisisa$var
- _aVariabel

Exempel på felaktiga variabel namn:

- 123myVariabel
- var
- 15

Använda en variabel

När du har skapat en variabel kan du sedan använda den med hjälp av namnet. Om vi nu ville skriva ut vårt nummer till konsolen kan vi göra så här:
console.log(myNumber);
Detta hade då skrivit ut 15.
Vi kommer till mer sätt att använda variabler i senare kapitel.

Ändra en variabel

Om du har skapat en variabel och sedan vill ändra den kan det se ut så här:

myNumber = 10;
När variabeln redan finns behöver man inte använda **let** ordet, man behöver bara namnet och den nya datan.
Eftersom javascript inte använder typer på samma sätt som andra språk kan du även ändra din variabel till en helt ny typ så här:
myNumber = "Nu är jag text"

Variabel typer

Nu vet du att javascript inte kräver att du skriver typen men alla variabler har fortfarande en typ som javascript håller koll på, den typen bestämmer javascript själv baserat på hur datan ser ut. Det finns ganska många typer för variabler men börja med att lära dig 3 stycken: **int(nummer), string(text), boolean(sant/falskt).**

- **Int** - Int är kort för Integer vilket är heltal på engelska. Alla heltal upp till 2147483647 räknas som **int.** Negativa tal är också **int.** Tal som inte är heltal är inte int, de är en annan typ som kallas float. Exempel: **let myInt = 20;**
- **String** - För att någonting ska bli en string måste det vara inom citattecken **"Detta är en string"**. Du kan även lägga siffor inom citattecken och så blir det en string istället för en int. Alla tecken inom citattecken blir en string, inte bara bokstäver. Exempel: **let myString = "string text här, 123!";**
- **Boolean** - Detta är en variabel som bara kan ha ett av två värde, true(sant) eller false(falskt). Detta kan vara bra att för kod brukar ofta kolla om någonting är sant eller falskt. Exempel: **let isThisTrue = true;**

Frivillig extra läsning

https://www.w3schools.com/js/js_variables.asp

https://www.w3schools.com/js/js_datatypes.asp

https://www.javascript.com/learn/variables

Funktioner

Vad är en funktion?

Du vet nu att en variabel är data som vi har gett ett namn så vi kan använda det senare, man kan säga att en funktion är javascript kod som vi ger namn så vi kan använda det senare.

En funktion tar in data, gör någonting med den och sedan ger tillbaka data. Ibland så kommer en funktion inte att ge tillbaka någon data, man har den bara för att göra något och får ingen data tillbaka.

Ibland kan du märka att funktioner kallas istället metoder(methods), det finns små skillnader på metoder och funktioner men du kan tänka på dem som nästan samma sak.

Skapa en funktion

Här är ett exempel på att skapa en javascript funktion som adderar 10 till ett nummer:

```
function addTen(myNumber) {
        let myResult = myNumber + 10;
        return myResult;
}
```

I detta exempel så är **myNumber** den data som funktionen tar in och **myResult** är den data som skickas ut. Man använder **retur** för att säga vad som ska skickas ut eller som "returneras". All kod i en funktion är inom dessa tecken som kallas “curly brackets”: **{kod här}**.

Använda en funktion

Låt oss säga vi vill använda den funktionen vi skapade ovan som heter **addTen**, det gör man så här:

```
addTen(5);
```

Det vi skriver inom parenteser är den data som vi skickar till funktionen. I vårt exempel ovan så hette det **myNumber**. Problemet med att använda funktionen så här är att den data vi får tillbaka kommer inte användas till något, om din funktion ger data så borde du oftast använda den på något sätt. Du kan till exempel skapa en variabel för datan, det ser ut så här:

```
let myVariable = addTen(5);
```

På detta sätt kommer datan från funktionen att sparas i en variabel och vi kan använda den senare. I vårt fall kommer vår variable ha värdet 15 eftersom vi tar 5 och lägger till 10.

Parameter

Den data som skickas till en funktion kallas antingen parameter eller argument. En funktion kan ha flera parameters/arguments. I vårt exempel ovan så är **myNumber** vår parameter. Du säger hur många parameter en funktion ska ha när du skapar den, du ger dem namn med ett komma mellan varje, så här:

```
Function myFunc(param1, param2, param3){
Code here
}
```

När du sedan vill använda funktionen så behöver du ha med lika många paramter som du hade när du skapade den. Så för att använda **myfunc** så kan du skriva

```
myFunc(15, 30, 45);
```

Då kommer param1 ha värdet 15, param2 ha värdet 30 och param3 ha värdet 45.

Javascript standard funktioner

Javascript har flera funktioner som redan finns i språket och som du kan använda i din kod. Du kommer lära dig flera av dessa senare i guiden och de används ofta.

Det finns väldigt många standardfunktioner och jag kan inte lista dem alla, men du kan känna igen att någonting är en funktion baserat på hur de ser ut. Som du lärde dig ovan används en funktion med sitt namn och parenteser så om du ser det här: **filter()** så vet du att det är en funktion även om du inte vet vad den gör. Du hade då kunnat gå till google och skriva **javascript filter function** och lärt dig mer.

Frivillig extra läsning

https://www.w3schools.com/js/js_functions.asp

https://javascript.info/function-basics

https://www.tutorialspoint.com/javascript/javascript_builtin_functions.htm

Integer(nummer)

I denna delen går vi in djupare på variabel typen som heter **int**. Som jag nämnde tidigare är detta heltal, positiva eller negativa, upp till ungefär 2 miljarder. Detta är en väldigt vanlig variabel typ som man kan använda för att sätta ett värde på något eller räkna något.

Aritmetik

Det är vanligt att du vill göra lite enkel matematik med dina nummer, det kan du göra så här:

- **Multiplikation:** 5 * 5
- **Subtraktion:** 5 - 5
- **Addition:** 5 + 5
- **Division:** 5 / 5

Detta kan du använda med dina variabler, det kan se ut så här:

```
let myNum = 5;
let  newNum = myNum + 10;
let anotherNum = newNum - myNum;
```

Du kan också göra ändring på en variabel:

```
let myNum = 5;
myNum = myNum + 10;
```

Du kan skriva detta på ett annat sätt:

```
let myNum = 5;
myNum += 10;
```

Detta ger samma resultat som ovan men kan vara snyggare eller snabbare.
Ett extra trick att kunna är ett snabbt sätt att öka med 1 eller minska med 1:
myNum++; //Detta ökar myNum med ett
myNum--; //Detta minskar myNum med ett
Detta kommer vara användbart i det framtida kapitel som heter "loops"

Du kanske märker att om man tar t:ex 5/2 så får man inte ett heltal, vilket betyder att det inte är en integer.
Det blir istället en typ som heter float. Detta kommer du inte märka eftersom javascript inte kräver att du skriver typer, men det är bra att veta att i bakgrunden är det en annan typ av variabel

Frivillig extra läsning

https://www.w3schools.com/Js/js_numbers.asp

https://www.w3schools.com/Js/js_arithmetic.asp

String(text)

Vad är en string

I variabler kapitlet lärde du dig att string är en av de vanligaste variabla typerna. String är text, alla bokstäver, siffror och andra tecken kan vara string. En string är alltid i citattecken **“string här”**, man kan också använda enkla citattecken istället för dubbla **‘också en string’**.

Hur många tecken är min string

Hur många tecken en string har kallas längd. Du kan få längden på din string genom att använda **length** på detta vis:
let exampleText = “här är lite text”
let textLength = exampleText.length;
I detta sammanhang är **length** någonting som kallas **property**, det är överkurs för nu, men du kan tänka på dem som ganska liknande till funktioner men du använder inte parentes och kan inte ge den någon data.Properties är kopplade till objekt vilket du kommer lära dig om senare.

Sätta ihop flera strings

När man använder nummer(int) så betyder plustecken(+) addition, men när du använder strings betyder det att du sätter ihop två strings till en. Här är ett exempel:
let hello = "hejsan";
let world = " världen";
let helloWorld = hello + world;
//värdet på helloWorld blir "hejsan världen"
Det är bra att komma ihåg att om du vill ha ett mellanrum behöver det vara med antigen i slutet av den första eller början av den andra strängen.

Att plustecken fungerar olika på olika typer kan bli förvirrande och leda till fel, ta detta som exempel:
let myNum = 10;
let myString = "10"
let result = myNum + myString
När du tittar på detta kanske du tror att resultatet blir 20, det är fel, resultatet blir “1010”. Det som händer är att om en av variablerna är en string och du använder plustecken så tror javascript att allt ska bli string. Så den tänker på det som text och inte siffror och sätter ihop dem.

Javascripts string funktioner

I funktioner kapitlet så nämnde jag att javascript har många standardfunktioner som du kan använda. Det finns många som används för att hantera strings. Här är en lista på några som är bra att kunna:

- toUpperCase() - omvandlar text till stora bokstäver
- toLowerCase() - omvandlar text till små bokstäver
- substr() - hämtar ut en del av en string
- replace() - ersätter en del av en string med en annan string

En full lista av javascripts string funktioner kan du hitta här:
https://www.w3schools.com/jsref/jsref_obj_string.asp

Du använder dessa funktioner genom att skriva variabel namnet, sedan en punkt och sedan funktion namnet. Här är ett exempel:
let myString = "Hejsan Världen"
let upperString = myString.toUpperCase();
//värdet på upperString blir "HEJSAN VÄRLDEN"
Denna funktionen **toUpperCase()** behöver ingen extra data(parameter) så därför är parenteserna tumma. Här är ett exempel på en funktion som behöver parameter:
let myString = "Hejsan Världen"
let extractedString = myString.substr(7)
//värdet på extractedString blir "Världen"

I detta exempel så hämtar vi ut en del av vår string, den parameter vi skickar med är var vi ska börja hämta från. Så vi börjar hämta från tecken nummer 7 och hämtar resten av texten så vi får “världen”. (Vissa funktioner börjar räkna från 0 istället för 1 vilket kan bli förvirrande)

Frivillig extra läsning

https://www.w3schools.com/jsref/jsref_obj_string.asp

https://www.freecodecamp.org/news/javascript-string-tutorial-string-methods-in-js/

https://javascript.info/string

If-Else

Vad är if-else

Det är vanligt i programmering att man vill köra någon kod beroende på ett vilkor. Man kanske vill köra någon kod om en siffra är högre än 100 eller om en text matchar ett specifikt ord.
Detta kan man göra med if-else, tänk på det som att säga om(if) denna saken händer gör detta, annars(else) gör en annan sak.
Här är ett exempel på if-else kod:

```
if(myNum == 10){
        console.log("Ditt nummer är 10");
}else{
        console.log("Ditt nummer är inte 10");
}
```

Med denna kod kollar vi om variabeln myNum har värdet 10, sedan skriver vi ut text beroende på om det är sant eller inte. Koden i parentesen är det vi kollar om det är sant och koden inom dessa **{}** är där koden är för om det är sant.
Du kanske undrar varför vi har två likhetstecken, det lär du dig om i nästa del

Jämföra värde

Det finns flera sätt du kan jämföra värde som du kan använda i din if-else kod, här är en lista med vilka tecken du använder:

- **==** Jämför om två värden är samma
- **===** Nästan samma som ovan men det kollar också om typerna är samma, det behöver du oftast inte använda
- **!=** Jämför om två värden **inte** är samma
- **>** Kollar om första värdet är större än andra värdet
- **<** Kollar om första värdet är mindre än andra värdet

- **>=** Kollar om först värdet är större eller lika med andra värdet

Anledningen till att vi använder två likhetstecken är eftersom ett likhetstecken är det vi använder när vi skapar variabler och annat så det är redan taget. Detta är viktigt att komma ihåg, om du använder bara ett i din if-else kod kan det bli helt fel.

Else if

Det är möjligt att du vill kolla om någonting är sant och sedan kolla om någonting annat är sant, detta kan du göra med **else if.** Här är ett exempel:

```
if(myNum > 10){
        console.log("Din siffra är över 10");
}else if(myNum < 10){
        console.log("Din siffra är under 10");
}else{
        console.log("Din siffra är exakt 10");
}
```

Här kollar vi först om vår variabel är över 10, sedan om det inte är sant så kollar vi i vår **else-if** om vår variabel är under 10. Om båda dessa två är falskt, då går vi till vår else och vi vet att siffran är exakt 10.

Denna koden hade kunnat vara fel eftersom det är möjligt att vår variabel inte ens är en siffra, då kommer vår **if** vara falsk och vi skriver att siffran är 10, vilket då är fel. Du hade kunnat fixa detta genom att lägga till en till **else if** som kollar om siffran är lika med 10.

Logical Operators

Ibland behöver man lite extra när man kollar om någonting är sant, då kan du använda dessa tecken:

- **&&** - Med detta kan du kolla om mer än en sak är sant, skriv dessa mellan två påstående och om både är sant så räknas det som sant, om bara ett är sant räknas det som falskt
- **||** - Dessa är liknande men här behöver bara ett påstående vara sant för att det ska räknas som sant.
- **!** - Detta gör att du får motsatt resultat till vad det var innan, så sant blir falskt och falskt blir sant.

Här är lite exempel:

Exempel1:
```
if(myNum > 10 && myNum < 30){
        console.log(“nummer är mer än 10 och mindre än 30”);
}
```

Exempel2:
```
if(myNum > 10 || mySecondNum > 10){
        console.log("åtminstonde ett av mina nummer är över 10 ”);
```

}
Exempel3: **if(!myNum > 10){**
console.log(“Mitt nummer är INTE mer än 10”);
}
Du kan kombinera dessa på många olika sätt och du kan ha med matematik du lärde dig innan. Du kan vara mycket kreativ. Man kan använda parenteser för att separate om det blir för mycket, t:ex
if((myNum > 10 && myNum < 30) || !(myNum == 10))

Frivillig extra läsning

https://www.w3schools.com/js/js_comparisons.asp

https://www.w3schools.com/js/js_if_else.asp

Arrays

Vad är en array

En array är en variabel typ, det är en variabel som istället för att hålla ett värde, håller en lista av värde. Här är två exempel på hur en array kan se ut i javascript:
let fruits = [“äpple”, “apelsin”, “päron”];
let numbers = [5, 10, 15, 20];
Alla värde i en array är inom fyrkantiga parenteser [] och det är ett komma mellan varje.
Du kan använda vilka variabeltyper som helst i din array, sträng, int och boolean fungerar alla bra.

Använda arrays

Alla värden i en array har en position, den börjar på 0 och går upp en i taget. I vår **fruits** array ovan så hade det blivit: äpple 0, apelsin 1, päron 2. Det kan skapa förvirring att positionen börjar på 0 istället för 1, detta är något du måste komma ihåg, det är lätt att glömma.
Du kan använda denna position för att få värdet, på detta sätt:
console.log(fruits[0]); //skriver ut äpple
console.log(fruits[2]); //skriver ut päron

Det är vanligt att du vill gå igenom alla värde i en array och göra något med varje, detta kommer du lära dig om i nästa kapitel “Loops”.
Du kan ändra värde i en array på samma sätt som en variabel:
fruits[1] = grapefrukt;
Nu är värde på position 1 grapefrukt istället för apelsin.

Array funktioner

Det finns en mängd inbyggda javascript funktioner som man kan använda på arrays. Här är en lista på några av de vanligaste

- push() - Lägger till ett värde till arrayen(på sista position)
- pop() - Tar bort värdet på sista positionen
- sort() - Sorterar en array(Fungerar bra på strings, är lite mer komplicerat för int)
- toString() - Omvandlar en array till en sträng

Array length

Precis som med strängar så kan du använda **length** för att få längden på en array. Detta är igen inte en funktion, det är en property, men du har inte lärt dig om property än.
Du kan använda detta om du till exempel vill skriva ut det sista värdet i din array. Men kom ihåg att positioner börjar på 0 vilket gör det svårare, se här:
Let fruits = ["äpple", "päron"];
console.log(fruits.length); //skriver ut 2
console.log(fruits[fruits.length]); //Ger ett felmedelande
I din fruits array här så är äpple 0 och päron 1. Position 2 finns inte, så det blir fel om du försöker skriva ut den. Det du behöver göra är att ta en mindre än längden, då får du sista värdet:
console.log(fruits[fruits.length-1]);

Frivillig extra läsning

https://www.w3schools.com/js/js_arrays.asp

https://www.w3schools.com/js/js_array_methods.asp

Loops

Vad är en loop

Loops är ett sätt att köra lite kod om och om igen till du tycker den är klar och ska sluta. Detta är väldigt vanligt att behöva göra med arrays. Det finns flera olika sorters loops i javascript men det finns två som är vanligast, "for loop" och "while loop".

- **for** - När du skapar en for loop så skapar först en int variabel som börjar på en siffra. Sedan säger du när den ska fortsätta loopa och till sist säger du vad siffran ska öka med varje loop. En enkel version hade kunnat vara, börja på 1, öka med 1 varje loop, sluta när du har kommit till 5.
- **while** - En while loop är ganska lik **if** kod som du lärde dig om innan. Du säger om(if) detta är sant, fortsätt loopen och när det inte är sant längre så slutar loopen.

for loop

För någon som aldrig sett en for loop innan kan den se konstig och förvirrande ut, jag visar ett exempel och förklarar efteråt:

```
//En for loop som skriver ut 0-9
for(let i = 0; i < 10; i++){
        console.log(i);
}
```

Du kan se att denna loopen är väldigt kort och kompakt. Som jag nämnde innan, behöver du göra 3 saker i din for loop och det är ett semikolon mellan varje sak du gör.

1. Skapa en int variabel som börjar på en siffra. I vårt exempel ovan så skapar vi en variabel som heter **i** och som har är siffran 0. Den måste inte heta **i** och den måste inte börja på 0.
2. Sedan säger du när loopen ska fortsätta. I vårt exempel säger vi att loopen ska fortsätta så länge vår variabel **i** är mindre än 10(i < 10). Denna loop kommer inte skriva ut 10 eftersom symbolen **<** är bara sant om det är under 10, inte om det är lika med. Du hade kunnat använda **<=** för att fixa det.
3. Till sist säger vi hur mycket i ska öka med varje loop. I vårt exempel vill vi att den ska öka med ett varje loop(i++), detta kanske du kommer ihåg från integer kapitlet. Det hade varit möjligt att öka med mer än 1 så här **i+=3**. Det såg du också i integer kapitlet.

Dessa tre saker ligger i en parentes och sedan lägger du koden som ska köras varje loop i dessa **{}**. I din loop kan du nu använda variabeln som du skapade, den heter **i** i vårt exempel. Det är ett vanligt namn på den när du gör en for loop.

Här är ett extra exempel som visar hur du kan använda for loop med en variabel:

```
let fruits =  ["äpple", "päron", "apelsin"];
for(let i = 0; i < fruits.length; i++){
        console.log("Positionen är " + i);
        console.log("Frukten är " + fruits[i])
}
```

I detta exempel så börjar vi på 0 och går igenom hela vår array genom att kolla om **i** är mindre än längden. I varje loop går vi igenom ett värde i vår array så att vår variabel har positionen för frukten. Vi använder **fruits[i]** för att få den frukten vi är på just då.

while loop

En while loop är lite enklare att förstå, vi kollar om någonting är sant och fortsätter tills det slutar vara sant, här är ett exempel:

```
//En while loop som skriver ut 0-9
let i = 0;
while(i < 10){
        console.log(i);
        i++;
}
```

I detta exempel är vår variabel skapad utanför vår while loop, till skillnad från en for loop. I varje loop kollar den om **i < 10** fortfarande är sant och slutar så fort det inte är sant. Strukturen på en while loop är precis som **if** kod, du behöver inte förstå något extra som i for loop.

En viktigt sak att komma ihåg med en while loop är att om du skriver fel så är det möjligt att den aldrig slutar loopa. Detta kallas en infinite loop(oändlig loop) och kommer att krascha hela ditt program. Detta hade hänt till exempel om vi tog bort raden **i++;** i ovan exempel. Då hade vår variabel aldrig ökat och hade alltid varit 0 och därför alltid mindre än 10. Var försiktig med detta.
Här är ett extra exempel med while loop och array:

```
let fruits =  ["äpple", "päron", "apelsin"];
let i = 0;
while(i < fruits.length){
        console.log("Positionen är " + i);
        console.log("Frukten är " + fruits[i])
}
```

Som du kan se så kan man använda de olika looparna för samma uppgift, det är upp till dig att bestämma vilken loop som passar bäst för vad du vill åstadkomma.

Frivillig extra läsning

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Loops_and_iteration

https://www.w3schools.com/js/js_loop_while.asp

https://www.w3schools.com/js/js_loop_for.asp

Javascript och HTML

HTML

För denna guide antar jag att du har en idé om hur HTML fungerar men om du inte har det kan jag sammanfatta snabbt. HTML används på alla hemsidor för att strukturera sidan. HTML är egentligen inte ett programmeringsspråk, det är ett språk som bara definierar element och var de ligger. Du använder taggar för att definiera element som text, bilder, rubriker, länkar och mer.
En tagg kan se ut så här:

```
<p>text här</p>
```

HTML element kan ha id och klasser vilket man använder för att identifiera eller gruppera dem. Du kommer använda id och klasser i din javascript kod för att bestämma vilka html element du vill påverka. När du använder Javascript kommer du ofta att jobba med html element så det behövs grundläggande kunskap om html.

Börja använda javascript på riktigt

Nu när du har lärt dig lite javascript så är det dags att faktiskt använda det i praktiken. För att använda Javascript behöver du ha en HTML fil redo. Du kan koppla din javascript kod till html filen på två sätt.

1. Script taggar - Du kan använda en specifik html tag för att ha med javascript kod i filen. Detta är bäst om du inte har så mycket kod. Den ser ut så här: **<script> //javascript kod här </script>**
2. Script länk - Du kan också länka till en Javascript fil från din html fil. Detta låter dig ha det separat vilket kan göra det enklare att hålla koll på. Om din fil heter myScript.js så länkar du så här:
   <script src="myScript.js"></script>

Hitta html element

För att hitta html element brukar man oftast använda id eller klasser, ibland kan man använda vilken tagg typ det är. Dessa funktioner är alla bra att veta:

- **document.getElementById(id)** - Letar efter ett html element som matchar id du ger som parameter, ger aldrig mer än ett element.
- **document.getElementsByClassName(name)** - Letar efter html element som matchar klass namn du ger, kan ge tillbaka flera html element som en array.
- **document.getElementsByTagName(name)** - Letar efter html element som matchar tag namn du ger, kan ge tillbaka flera html element som en array.

Du kan sedan sätta dessa i en variabel och om det är en array kan du behöva loopa igenom och göra något med varje html element.

Ändra html element

Alla html element har en stor mängd attribut som man kan påverka med javascript. Det kan vara allt från text, storlek på text, länken till en bild, färgen på en rubrik och mycket mer. Olika sorters element kan ha olika sorters attribut. Du kan ändra ett html element antigen direkt från en av funktionerna ovan eller spara det till en variabel och ändra från där. Du gör detta med en punkt och sedan namnet på attributet, här är två exempel:

```
//använder funktionen direkt
document.getElementById("myElem").height = 200;
//använder en variabel
let myElem = document.getElementById("myElem");
myElem.height = 200;
```

Både dessa två sätten kommer att göra samma sak, det handlar bara om du vill ha en variabel eller inte, vilket du oftast vill. Det finns väldigt många attribut och jag kan inte lista alla, men här är några viktiga:

- **innerHTML/innerText/textContent/value** - Detta är 4 olika attribut som ger väldigt liknande data. De ger alla texten som ett element har på ett eller annat sätt. Du behöver inte veta exakt vad alla gör, jag brukar oftast använda innerHTML och value.
- **children -** Ger dig alla "children" av html elementet, om du kan html vet du vad detta är. Det är andra element som typ ligger under detta element
- **id/class** - Du kan ändra id och klass också om du vill
- **src -** Används ofta på bilder för att få länken till bilden
- **href** - Används för länkar för att få var länken tar dig

Ändra Stil

Vissa attribut ligger under ett stil attribut så för att ändra dessa måste du gå igenom stil attributet först. Här är ett exempel:
//ändra färg på html element
let myElem = document.getElementById("myElem");
myElem.style.color = "red";
Detta är oftast saker du hade ändrat med CSS, det kan vara svårt att komma ihåg vilka saker räknas som stil men Google är din vän här.

Frivillig extra läsning

https://www.w3schools.com/tags/ref_attributes.asp

https://www.w3schools.com/js/js_htmldom.asp

https://www.w3schools.com/js/js_htmldom_document.asp

Events

Vad är Events

Ofta vill du att Javascript kod ska köras när någonting specifikt händer, då behöver du använda Events(händelser). Några av dessa event kan vara: sidan laddas in, användaren trycker på en knapp, ett formulär fylls i, användaren trycker på tangentbordet.

Kod Exempel

Det finns några olika sätt att skriva koden som hanterar events, men jag tror den du kommer se mest ser ut så här:
let myButton = document.getElementById("myButton");
function knappFunktion(){

```
    console.log("Någon tryckte på knappen");
}
myButton.addEventListener("click", knappFunktion);
```

I denna kod så skapar vi en variabel som håller en html button med id "myButton". Vi använder en funktion som heter **addEventListener()**, denna funktion säger till javascript att den ska kolla efter ett event på detta html element. Den parameter denna funktion behöver är vilken typ av event den ska kolla efter och en funktion som ska köras när det händer.

I detta exempel så har vi skapat en funktion tidigare som heter **knappFunktion** och det är den som kommer köras, det finns ett annat sätt att skriva detta om du inte redan har en funktion:

```
let myButton = document.getElementById("myButton");
myButton.addEventListener("click", new function(){
    console.log("Någon tryckte på knappen");
});
```

Denna kod kommer att göra exakt samma sak men vi skapar en funktion utan ett namn som bara används för just detta. Detta sättet att skriva är ganska vanligt.

Det går också att göra samma sak men att göra det i ditt html element, det ser ut så här:

```
<button onclick="KnappFunktion()">Klicka mig</button>
```

Detta fungerar lika bra, det handlar om hur du vill göra det och tycker är enklast att hålla koll på.

Man vill ofta att någon kod ska hända när sidan har laddats, det gör man så här:

```
window.addEventListener("load", myLoadFunction);
```

Frivillig extra läsning

https://www.w3schools.com/js/js_events.asp

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events